Object File Comparison and Record Types in UBROF Files from IAR C/C++ Compilers

Technical Note

1 Introduction

1.1 Revision History

Revision	Date	Author	Modification
A	2005-04-07		Initial revision

1.2 Purpose

It is sometimes necessary to compare the contents of two object files in order to determine if there are any significant differences between them. For various reasons, differences occur that do not affect the actual application code. This document describes which record types in a UBROF file that can affect the actual application code, and which record types that only give supplementary information used for debugging or other purposes.

1.3 Assumptions

Output from the list-object-code command in XLIB is used as a basis for the comparison.

1.4 Open Issues

-

2 Categorization

UBROF record types can be divided into two main categories:

- Record types that carry information that can affect the final code of the application
- Record types that carry supplementary information, such as debug information or information about the build process.

This document provides a list of all record types in each category, with a brief comment about the purpose for which the record type is used.

Note that some debug information (in particular, call frame information) is considered essential, because it is needed to provide basic debugging support in a application that mixes modules with debug information and modules *without* debug information. This kind of debug information is output by the compiler even when it is not configured to include debug information.

2.1 Record Types Affecting the Application

Many of these record types cannot occur in current versions of UBROF, but they are all listed here for completeness.

The record types marked “Static overlay” can also occur in UBROF files for products that are not using the static overlay features. In that case they only carry debug information.

Some symbols in a UBROF file exist only to carry debug information. In that case differences in, for instance, T_SYMBOL_DEF records for those symbols have no effect on the final application. One example of this is that symbols for a fully inlined function can be present in a UBROF file to provide a point of reference for debug information about that function. This only occurs when the compiler is configured to produce debug information.

T_SCOPED_NAME and T_TYPE records carry information about names and types in UBROF, starting with version 8. Whenever particular T_SCOPED_NAME records are referenced from T_SYMBOL_DEFx records, they can affect the final application. In the case of some C++ names, like overloaded functions, T_SCOPED_NAME records can refer to T_TYPE records. In that case these records too can affect the final application.

The contents and order of the following record types can affect the final application code:

Record Type	Purpose
T_ABS_16	Code or data bytes
T_ABS_24	Code or data bytes
T_ABS_32	Code or data bytes
T_ABS_8	Code or data bytes
T_ABS_BLOCK	Code or data bytes
T_ABS_VAR	Code or data bytes
T_AND	Relocation
T_BNOT	Relocation
T_CHECK	Relocation
T_CHECK_LIBRARY	Link-time checking
T_CHECK2	Relocation
T_COPY	Relocation
T_DATE	Relocation
T_delete_tos	Relocation
T_DIV	Relocation
T_END_ABS	Program start address
T_END_REL	Program start address
T_END_REL1	Program start address

T_EQ	Relocation
T_ERROR	Relocation
T_EXT_24	Relocation
T_EXT_32	Relocation
T_EXT_DEF	Symbol handling
T_EXT_FI_8	Relocation
T_EXT_N16	Relocation
T_EXT_N8	Relocation
T_EXT_NPCREL	Relocation
T_EXT_P16	Relocation
T_EXT_P8	Relocation
T_EXT_PPCREL	Relocation
T_EXT_S86	Relocation
T_GE	Relocation
T_get_byte	Relocation
T_GT	Relocation
T_HWRD	Relocation
T_IOR	Relocation
T_KEY_VALUE	Link time checking
T_LAND	Relocation
T_LDEF_FN_51	Static overlay
T_LE	Relocation
T_LI0_16	Relocation
T_LI0_8	Relocation
T_LI1_16	Relocation
T_LI1_8	Relocation
T_LI2_16	Relocation
T_LI2_8	Relocation
T_LI3_16	Relocation
T_LI3_8	Relocation
T_LI4_16	Relocation
T_LI4_8	Relocation
T_LI5_16	Relocation
T_LI5_8	Relocation
T_LOAD_IND	Relocation
T_LOC_ABS	Symbol handling
T_LOC_REL	Symbol handling
T_LOR	Relocation
T_LSL	Relocation
T_LSR	Relocation
T_LT	Relocation
T_LTEST	Relocation
T_LWRD	Relocation
T_LXOR	Relocation
T_MINUS	Relocation
T_MOD	Relocation
T_MUL	Relocation
T_NE	Relocation
T_NEG	Relocation
T_NOT	Relocation
T_ORG	Segment part handling
T_ORG_ABS	Segment part handling
T_ORG_REL	Segment part handling
T_ORG_REL1	Segment part handling
T_PLUS	Relocation
T_POP_16	Relocation

T_POP_24	Relocation
T_POP_32	Relocation
T_POP_8	Relocation
T_PRM_ARG_BL	Relocation
T_PRM_ARG_BL51	Relocation
T_PUB_ABS	Symbol handling
T_PUB_REL	Symbol handling
T_PUSH_ABS	Relocation
T_push_DAY	Relocation
T_PUSH_EXT	Relocation
T_push_HOUR	Relocation
T_PUSH_IFN	Relocation
T_PUSH_IFN51	Relocation
T_push_MINUTE	Relocation
T_push_MONTH	Relocation
T_PUSH_PCR	Relocation
T_PUSH_PRM_ARG_BL	Relocation
T_PUSH_REL	Relocation
T_push_SECOND	Relocation
T_PUSH_SFB	Relocation
T_PUSH_SFE	Relocation
T_push_YEAR	Relocation
T_QUEST	Relocation
T_QUEST_VAR	Relocation
T_REF_FN	Relocation
T_REF_FN51	Relocation
T_REF_FNT	Relocation
T_REL_24	Relocation
T_REL_32	Relocation
T_REL_FI_8	Relocation
T_REL_N16	Relocation
T_REL_N8	Relocation
T_REL_NPCREL	Relocation
T_REL_P16	Relocation
T_REL_P8	Relocation
T_REL_PPCREL	Relocation
T_REL_S86	Relocation
T_SCOPED_NAME	Symbol handling (and type information)
T_SEG	Segment part handling
T_SEG_ABS	Segment part handling
T_SEG_COM	Segment part handling
T_SEG_INFO	Segment part handling
T_SEG_REL	Segment part handling
T_SEG_STK	Segment part handling
T_SFB_24	Relocation
T_SFB_32	Relocation
T_SFB_N16	Relocation
T_SFB_N8	Relocation
T_SFB_NPCREL	Relocation
T_SFB_P16	Relocation
T_SFB_P8	Relocation
T_SFB_PPCREL	Relocation
T_SFE_24	Relocation
T_SFE_32	Relocation
T_SFE_N16	Relocation
T_SFE_N8	Relocation

T_SFE_NPCREL	Relocation
T_SFE_P16	Relocation
T_SFE_P8	Relocation
T_SFE_PPCREL	Relocation
T_SHL	Relocation
T_SHR	Relocation
T_SRC_CALL	Static overlay
T_stack_error	Relocation
T_STORE_IND	Relocation
T_SUB_DEF_FRAME_SIZE	Static overlay
T_SUB_DEF_FRAME_SIZE1	Static overlay
T_SUB_DEF_FUNC	Static overlay
T_SUB_DEF_STAT_ARGS	Static overlay
T_SUB_DEF_STAT_LOCS	Static overlay
T_SUB_DEF_USEAGE	Static overlay
T_SUB_DEF_XFUNC	Static overlay
T_SWAP	Relocation
T_SYMBOL_DEF	Symbol handling
T_SYMBOL_DEF1	Symbol handling
T_SYMBOL_DEF2	Symbol handling
T_TYPE	Symbol handling for some C++ symbols and type information
T_UGE	Relocation
T_UGT	Relocation
T_ULE	Relocation
T_ULT	Relocation
T_XDEF_FN_51	Static overlay
T_XOR	Relocation

2.2 Supplementary Record Types

The content and order of the following record types cannot affect the final application code:

Record Type	Purpose
T_ATTR_NAME	Type information
T_AUX	Build information
T_AUX_1	Build information
T_BEG_FILE	File structure
T_BEG_FILE1	File structure
T_BEG_LIB	File structure
T_BEG_PGM	File structure
T_BLOCK_BEGIN	Debug information
T_BLOCK_BEGIN1	Debug information
T_BLOCK_BEGIN2	Debug information
T_BLOCK_BEGIN3	Debug information
T_BLOCK_END	Debug information
T_CALL_FRAME_INFO	Call stack information
T_COUNT	File structure
T_DEBUG	Build information
T_DEF_PTR_TYPES	Type information
T_END	File structure
T_FILE	Debug information
T_FILE_REF	Debug information
T_FUNC_BEGIN	Debug information
T_FUNC_BEGIN1	Debug information
T_FUNC_BEGIN2	Debug information

T_GRP	Debug information
T_GRP_REF	Debug information
T_MACRO	Debug information
T_MEM_INFO	Type information
T_NEW_FILE	Debug information
T_RESOURCE_TUPLE	Debug information
T_SIZE_TYPE	Type information
T_SOURCE_LINE_SEQUENCE	Debug information
T_SOURCE_RANGE	Debug information
T_SOURCE_RANGE1	Debug information
T_SRC_STAT	Debug information
T_STATEMENT_INFO	Debug information
T_SUB_DEF_OBJ_ATTR	Debug information
T_UBROF_EOF	File structure
T_VERSION	Build information